Escuela Politécnica Superior

How2Grasp: Dataset sintético de agarres virtuales

Máster Universitario en Automática y Robótica

Trabajo Fin de Máster

Autor:
Ángel Manuel Gama García
Tutor/es:
Vicente Morell Giménez
José García Rodríguez
Pablo Martínez González

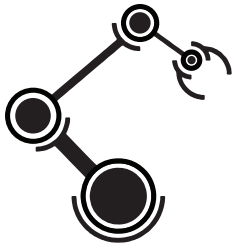
Julio 2021

Universitat d'Alacant
Universidad de Alicante

How2Grasp: Dataset sintético de agarres virtuales

Autor
Ángel Manuel Gama García

Tutor/es
Vicente Morell Giménez
Departamento de Física, Ingeniería de Sistemas y Teoría de la Señal
José García Rodríguez
Departamento de Tecnología Informática y Computación
Pablo Martínez González
Departamento de Tecnología Informática y Computación



Máster Universitario en Automática y Robótica

Escuela Politécnica Superior

Universitat d'Alacant
Universidad de Alicante

ALICANTE, Julio 2021

Preámbulo

En este trabajo se ha desarrollado un datasetcompleto de datos sintéticos basado en agarres sobre objetos para su posterior uso en proyectos de robótica y visión artificial que utilizan técnicas de *Deep Learning*.

Los objetos con los que se llevarán a cabo los agarres forman parte del dataset YCB. El dataset dataset propuesto proporciona información de los puntos de contacto de cada uno de los dedos individualmente. Además, se va a obtener un mapa de calor que unifique todos los contactos de los cinco dedos y se diferencien asignándole un color específico. Para cumplir este objetivo tan definido se va a emplear tecnología de realidad virtual mediante el uso de las gafas Oculus Quest y el motor gráfico Unreal Engine 4 (UE4) sobre el que se va a ejecutar el entorno de simulación.

Agradecimientos

Agradecer a todos los profesores que he tenido a lo largo del máster sobretodo por su atención en este año atípico. Además dar las gracias a mis tutores de este TFG por la ayuda en todo momento, en especial a Pablo por la paciencia y dedicación. Por último agradecer las personas que han llevado a cabo los diferentes proyectos que dan base a How2Grasp.

A mi familia, causa de todos y cada uno de mis avances.

Everybody is a genius. But if you judge a fish by its ability to climb a tree, it will live its whole life believing that it is stupid.

Albert Einstein.

Índice general

Índice de figuras

Lista de Acrónimos y Abreviaturas

CNN	Convolutional Neural Network.
EOAT	End Of Arm Tooling.
FPS	Frames Per Second.
GQ-CNN	Grasp Quality Convolutional Neural Networks.
HDR	High Dynamic Range.
HSR	Human Support Robot.
IA	Inteligencia Artificial.
MPPH	Mean Picks Per Hour.
RA	Realidad Aumentada.
RGB-D	Red Green Blue - Depth.
RM	Realidad Mixta.
RV	Realidad Virtual.
TFG	Trabajo Final de Grado.
TFM	Trabajo Final de Máster.
UA	Universidad de Alicante.
UE4	Unreal Engine 4.
UWS	University of West Scotland.
YCB	Yale-CMU-Berkeley.

1 Introducción

La Realidad Virtual (RV) y aumentada (RA) están alcanzando cuotas muy altas de popularidad. Esto se debe, principalmente, a la gran cantidad de aplicaciones que tiene en numerosos sectores como puede ser la medicina, entretenimiento, educación o robótica. Debido a unos requerimientos de realismo cada vez más exigentes de las aplicaciones de RV/RA desarrolladas, surge la necesidad de una mejora constante en las tecnologías. Esta mejora viene impulsada con las generaciones de nuevos dispositivos de empresas del sector como pueden ser Oculus o HTC.

Un factor importante de los métodos que emplean realidad virtual es la forma en la que se interacciona con los objetos, siendo la característica abordada en este Trabajo Final de Máster (TFM) el método de agarre. Los seres humanos somos capaces de intuir los puntos de agarre más estables de un objeto eficientemente, teniendo en cuenta parámetros como forma, peso y la posición final para trasladarlo, cosa que hacemos de forma natural. En este TFM se va a completar un dataset sintético de agarres de los objetos del dataset Yale-CMU-Berkeley (YCB) (Calli y cols., 2015) descrito en la Sección 2.3.2.1, haciendo uso de gafas de realidad virtual (RV) Oculus Quest, que añaden una función de seguimiento de manos (o *handtracking*). Esta funcionalidad permite utilizar nuestra propia mano para realizar el gesto de agarre, consiguiendo así plasmar esa intuición natural de agarre en los datos que se generarán en el entorno virtual. Más adelante veremos qué significa y qué implicaciones tiene que el dataset esté compuesto por datos sintéticos.

Cuando se trabaja con robots, la decisión de cómo coger o trasladar los objetos es toda una área de estudio, siendo el agarre inteligente un objetivo real. El robot, dependiendo del dispositivo de agarre del que disponga, y del objeto que quiera coger, deberá de comportarse de una forma u otra. A lo largo de este TFM se va a tratar de relacionar un grasping robótico con realidad virtual en un entorno creado en Unreal Engine 4 (UE4).

Con el avance de los algoritmos de aprendizaje profundo que dominan las soluciones de visión artificial, y más en concreto la clasificación y manipulación de objetos, los datos sintéticos son cada vez más populares, esto es debido a su gran utilidad ya que nos permiten generar datos de una forma más rápida, y anotarlos automáticamente. Consisten básicamente en información específica generada haciendo uso de simulaciones a través de un ordenador, y tratan de aproximarse a la información del mundo real.

Este proyecto final[1] trata de ampliar y profundizar el TFG *UnrealHandGrasp* (Mulero Pérez, 2021), en el cual se desarrolla un sistema de agarre de objetos en realidad virtual en UE4, haciendo uso del *handtracking* proporcionado por las Oculus Quest. En el proceso, se llevarán a cabo una serie de objetivos que serán detallados en la sección 1.2

A lo largo del documento se van a tratar de explicar a fondo las tecnologías en relación a las técnicas empleadas, dejando claro los objetivos y la forma en la que se han cubierto.

[1] https://github.com/3dperceptionlab/unrealhandgrasp/tree/dataset

1.1 Motivación

Para tratar la motivación de este proyecto se va a dividir en dos tipos, primero se va a comentar la motivación en relación al porqué de la utilización de esta técnica, y posteriormente se abordará la motivación personal.

Lo que nos lleva a crear un dataset sintético de agarres de objetos es, principalmente, su posterior aplicación en el entrenamiento de modelos de Deep Learning, con la esperanza de que el sistema entrenado con datos obtenidos de simulaciones luego se comporte correctamente en el mundo real. Es por esto que se tratará de crear el dataset lo más fiel a la realidad posible.

Así, el trabajo consiste en extender el sistema de agarres implementado en UE4 para extraer la información necesaria para conformar el dataset de una forma rápida y automática. También se adaptará el sistema de agarre a otro tipo de manipulador, como puede ser la pinza de un robot, para extraer la misma información de agarres.

En cuanto a la motivación personal, tras la finalización del grado en Ingeniería Robótica en la Universidad de Alicante (UA), prosigo mis estudios en esta misma universidad con el máster en Automática y Robótica. Durante este año de máster me surge la oportunidad de realizar un doctorado codirigido por tutores de dos universidades, UA y University of West Scotland (UWS), siendo José García uno de ellos, y ofreciéndome además la opción de llevar a cabo mi TFM, junto a Vicente MOrell y con la colaboración de Pablo Martínez.

Algo que me define es la motivación de aprender cosas nuevas de diferentes ramas y formarme en nuevos ámbitos para ser lo más multidisciplinar posible, es por esto que este proyecto es perfecto para obtener conocimientos sobre realidad virtual y el software Unreal Engine, un campo que me ha parecido interesante desde el primer momento.

1.2 Objetivos

Los objetivos de este proyecto, teniendo en cuenta los intereses principales, se pueden dividir en dos subobjetivos.

El primer subobjetivo se basa en conformar un dataset con agarres sobre todos los objetos del dataset YCB haciendo uso de un entorno creado en UE4. El segundo subobjetivo consiste en la implementación de una lógica que permita extraer los agarres con un formato de mapa de calor. Se plantea obtener estos mapas de calor independientemente para cada dedo, lo cual permitiría contar con aún más información que podría resultar útil.

Para la realización de estos subobjetivos clave se deberán realizar diferentes tareas:

1. Obtención de los agarres como mapas de calor.
    - Extensión de la implementación en UE4 de la lógica para la extracción de los contactos como mapas de calor.
    - Obtención de un mapa de calor por dedo para contar con dicha información de manera disgregada.
    - Creación de un script en Python capaz de abrir cada uno de los contactos de los dedos y unificarlos en un solo mapa de calor en el que se pueda diferenciar cada uno de los dedos y que represente un agarre individual.
2. Creación del dataset.

- Preparación de un entorno de simulación en el que aparezcan todos los objetos del dataset YCB.
- Investigación en relación a formatos de datasets e implementación de la lógica que almacene los datos en UE4.

1.3 Estructura del documento

Para exponer la información relacionada con este trabajo se va a dividir el documento en distintas secciones y subsecciones, con tal de organizarlo correctamente.

Una vez introducido brevemente el trasfondo del TFM en esta misma sección, se ha comentado la motivación tanto general como personal y los dos objetivos principales. En la siguiente sección se da paso al estado del arte, describiendo los conceptos clave, proyectos previos que suponen la base de este, y datasets que guardan relación relevante con el que vamos a crear, que ha sido denominado *how2grasp*. Posteriormente llegamos al apartado de metodología, donde se hace referencia, principalmente, al hardware y software empleados. Una vez planteado el contexto que rodea este TFM, pasamos a explicar el desarrollo llevado a cabo en este, apartado que incluirá información y resultados de cada uno de los objetivos marcados y realizados. Por último, se hace una breve conclusión donde se expone los objetivos logrados exitosamente y las posibles propuestas de mejora futuras.

2 Estado del arte

En este apartado se van a examinar las técnicas relacionadas con el proyecto, presentando definiciones de realidad virtual y aumentada, y revisando antecedentes que muestran relación con nuestro proyecto. De esta forma se trata de poner en situación las tecnologías e introducir el estado de estas.

En el campo de la robótica, una de los aspectos más relevantes a considerar es la herramienta del extremo del robot, la cual se identifica como End Of Arm Tooling (EOAT). Esta herramienta se define previamente en base a la aplicación deseada, de forma que el robot sea capaz de llevar a cabo la tarea objetivo con la herramienta, como puede ser una pinza para transportar objetos en una aplicación básica de *pick and place*. A lo largo del estado del arte revisaremos datasets centrados en robótica, como Dexterity Network (sección 2.3.2.5), o bien otros que, adaptándose a las necesidades tecnológicas, tratan la robótica como un tema central en su proyecto como es el caso de Dex-YCB (sección 2.3.2.2).

2.1 Conceptos clave

En esta sección presentamos algunos conceptos importantes para entender los proyectos revisados en secciones posteriores.

2.1.1 Realidad virtual, aumentada y mixta

Se trata de tecnologías en continuo desarrollo y relativamente nuevas, por lo que es común mezclar y confundir los términos Realidad Virtual (RV) y Realidad Aumentada (RA). Esto es debido principalmente a que ambos términos muestran elementos diseñados por un ordenador que no existen en la vida real. Puesto que ambas tecnologías están cada vez más presentes en el mundo cotidiano, se tratará de aclarar su diferencia.

2.1.1.1 Realidad virtual

El término realidad virtual puede ser el más conocido entre los comparados previamente, esta tecnología se sustenta sobre dispositivos como gafas especialmente diseñadas para brindarnos tal comportamiento, que actualmente están en un momento de auge con un desarrollo constante. La realidad virtual proporciona una experiencia inmersiva gracias a las gafas que incorporan una pantalla, auriculares, mandos y diferentes añadidos. Es decir, crea una realidad imaginaria en la que las posibilidades son prácticamente infinitas. Algunos ejemplos de la capacidad de esta tecnología pueden ser simulaciones interactivas en 3D, vídeos grabados de forma que permiten al usuario una vista de 360 y audio inmersivo.

Aunque se usa en multitud de campos, el más potenciado en los últimos años es en la industria del entretenimiento, con cada vez más títulos de videojuegos incorporan características

de realidad virtual. El crecimiento en este ámbito es algo lógico debido a la fácil accesibilidad a casi cualquier tipo de público.

Se dice que el primer simulador de la historia de RV fue *Sensorama*. Se trataba de una máquina multisensorial conocida por ser de los primeros sistemas inmersivos, fue creada a finales de los años 50 y no nació en relación a los ordenadores o videojuegos, sino para ofrecer una experiencia cinematográfica completa.

Figura 2.1: Imagen primer simulador VR de la historia. (FERNÁNDEZ, s.f.)

En la imagen 2.1 se puede observar lo que ofrecía este sistema tan innovador en la época, algo totalmente revolucionario y que sin duda marcó el inicio de algo increíble en la actualidad.

2.1.1.2 Realidad aumentada

El término realidad aumentada, en cambio, hace uso del mundo real para superponer información digital sobre elementos reales, es decir, se puede decir que trata de añadir o mejorar la realidad con elementos inexistentes.

Algunos ejemplos de RA son: en el mundo de la moda la capacidad de visualizar las prendas virtuales puestas sobre ti, en el mundo real. O en los videojuegos el conocido juego de Pokémon GO, en el cual, mediante la cámara del móvil, eres capaz de ver criaturas creadas por el videojuego e interaccionar con ellas superpuestas al mundo real.

2.1.1.3 Diferencias

Una vez se han introducido los términos RV y RA procedemos a explicar las principales diferencias en este apartado. Como se puede observar hay bastantes diferencias a simple vista que pueden distinguir claramente ambas tecnologías, a continuación vamos a proceder a tratar de comentar las principales (Zapatero, s.f.).

- Grado de inmersión: cuando hablamos de RV la inmersión es completa en un mundo virtual, mientras que en la aumentada todo lo que se observa está sobre el entorno real.
- Sistema de visionado: Aunque no es obligatorio el sistema de gafas en RV, es algo prácticamente imprescindible, en cambio la RA no siempre son necesarias debido a la diferencia del grado de inmersión ya comentada.
- Entorno: En RV los objetos parecen reales de cara a la percepción del ser humano, en cambio en la RV, aunque se acercan a la realidad, normalmente hay una diferencia notable con respecto al mundo real.
- Visualización: En RA se superponen los objetos virtuales al mundo real, mientras que en RA se visualiza un mundo completamente virtual.

2.1.1.4 Realidad mixta

Con el avance de las tecnologías de Realidad Virtual y Realidad Aumentada cada vez se habla más de la aparición de la Realidad Mixta (RM), la última en llegar. Es un término algo complejo y nuevo, por lo que hay bastante desconocimiento acerca de esta. La RM es una combinación de ambas, lo que supone la fusión del mundo físico con el virtual, es decir, trata de llevar el mundo real al mundo virtual. La idea es la generación del modelo 3D de la realidad y sobre él superponer información virtual. Esta tecnología lleva al extremo el grado de inmersión, ya que ahora la información virtual va a estar afectada completamente con la real, algo del mundo real puede obstaculizar, tapar, afectarle en la iluminación a un objeto virtual, y viceversa.

2.2 Antecedentes

En este apartado se va a comentar los proyectos previos a *how2grasp*, los cuales marcan la base del entorno utilizado y son, en orden cronológico:

1. UnrealROX: Su objetivo es la creación de un entorno virtual hiperrealista. Trabajo relacionado con la generación sintética de datos, es el framework a partir del cual se han desarrollado el resto de proyectos de datos sintéticos. Posteriormente se ha desarrollado una mejora denominada UnrealROX+.
2. UnrealGrasp: Creación de un sistema de agarres robusto en tiempo real en un entorno virtual.
3. UnrealHandGrasp: Mejora del sistema de agarres haciendo uso del handtracking de las gafas de realidad virtual Oculus Quest.

2.2.1 UnrealROX

Este proyecto marca el inicio de una serie de trabajos relacionados con entornos realistas empleando realidad virtual, generación de datos sintéticos, creación de datasets, etc. Consiste en la creación de un entorno virtual extremadamente fotorrealista empleado para la generación de datos sintéticos de aplicacion, principalmente, en tareas de visión robótica. Es un entorno construido sobre UE4 con detalles característicos que acerquen el entorno a la realidad y permita la adquisición de datos de una forma más rápida y automatizada, de esta forma se podrá dotar a algoritmos de IA, que requieren grandes cantidades de datos para se entrenados, con datos de calidad que se asemejan a los adquiridos del mundo real.

El proyecto se ha planteado de la forma más genérica posible, de manera que soporta diferentes gafas de realidad virtual (ha sido probado exitosamente con las gafas Oculus Rift, Oculus Quest y HTC Vive). Proporcionando la posibilidad de controlar el entorno mediante los mandos de las gafas RV o incluso el teclado.

En cuanto a los robots y humanoides integrados en las pruebas, son los siguientes: Pepper y Mannequin de UE4 (Figura 2.2). Los robots están representados por la malla que los modela, la lógica de control y movimiento, las animaciones que lanza y el sistema de grasping. La forma de trabajar con los robots es uno de los puntos más importantes, ya que va a ser lo que defina el sistema de interacción con el entorno.

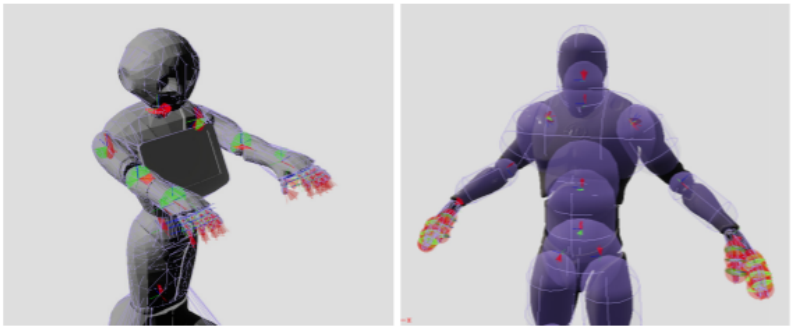


Figura 2.2: A la izquierda robot Pepper, a la derecha robot Mannequin (Martinez-Gonzalez y cols., 2019).

UnrealROX (Martinez-Gonzalez y cols., 2019) presenta una gran cantidad de posibles aplicaciones de los entornos hiperrealistas. Este mismo entorno de generación de datos fue usado para generar un dataset orientado a robótica asistencial denominado The RobotriX (Garcia-Garcia y cols., 2018). Este dataset consiste en datos de interiores extremadamente fotorrealistas obtenidos por agentes robóticos interactuando con objetos en un mundo simulado en UE4. De esta forma, se podrán usar estos datos en una amplia variedad de problemas de visión robótica.

2.2.2 UnrealGrasp

Una vez se ha descrito el proyecto base para la generación de datos de datos sintéticos hiperreaalistas y su aplicación para crear un dataset concreto de gran escala, pasamos a edescribir UnrealGrasp (Oprea y cols., 2019). Este proyecto se centra en un sistema de agarres

robusto en tiempo real, buscando la creación de un sistema de agarre visualmente realista en un entorno creado también en UE4, para manipular e interaccionar con objetos. Este proyecto constituye el comienzo de una serie de trabajos para el desarrollo de un sistema de agarres haciendo uso de gafas de realidad virtual. Los principales puntos definidos en este trabajo se centran en la creación de un sistema simple y modular, flexible, robusto y visualmente realista. Se hace un estudio de los tipos de grasping y se prueba el sistema creado con objetos del dataset YCB (Calli y cols., 2015).

2.2.3 UnrealHandGrasp

Tras asentar el sistema de agarres en UnrealGrasp (Sección 2.2.2), se ha seguido trabajando y mejorando sus capacidades. En el TFG (Mulero Pérez, 2021), se hace uso del handtracking implementado en el framework de Oculus para interaccionar con los objetos del entorno virtual creado en UE4. De esta forma, ahora se usarán únicamente las manos para los agarres y no los mandos de las gafas como se hacía previamente.

Implementa un algoritmo que se encarga de decidir cuándo mover y bloquear los dedos de la mano virtual. De manera que, cuando cerremos la mano en el mundo real se agarre un objeto en el mundo virtual con los dedos bloqueados donde ha tenido lugar el contacto con el objeto. De igual forma cuando se abre la mano, se suelta el objeto.

2.3 Datasets

El término anglosajón dataset hace referencia a un conjunto de datos tabulados, donde cada columna hace referencia a una variable y cada fila a un dato. Los datasets usualmente son utilizados como base para entrenar un algoritmo de inteligencia artificial y que éste sea capaz de aprender patrones característicos a partir de estos datos.

2.3.1 Tipos de datasets

Hay infinidad de datasets dependiendo de su finalidad. Cuando hablamos de robótica los más comunes son relacionados con la locomoción del robot, visión o con vehículos móviles entre otros muchos campos.

Cuando hablamos de datasets nos imaginamos un único archivo con información, y esto no es del todo correcto ya que hay diferentes tipos, los cuales vamos a dividir y definir en este apartado teniendo en cuenta origen y formato(*¿Qué son los Datasets y dónde conseguirlos?*, s.f.).

- Archivo: el más usual durante el aprendizaje debido a la seguridad y rapidez a la hora de emplearlos. Consiste en un fichero independiente en el que se encuentra la información.
- Carpeta: consiste en la suma de diferentes datasets almacenados en una carpeta interconectados entre sí. Comparten formato como por ejemplo: .csv, .mif o .dxf.
- Base de datos: también es un fichero, pero en este caso tienen formatos específicos diseñados para programas puntuales.
- Web: formado por los datos almacenados en un sitio web.

2.3.2 Datasets relacionados

Se ha realizado un estudio de los diferentes datasets actuales en relación al grasping tanto para estimación de poses de mano humana como para EOAT robóticos. De esta forma se va a tratar de analizar el estado de estos introduciendo los más acordes a nuestra aplicación. Se va a hacer especial hincapié en el proyecto ContactDB (sección 2.3.2.3), el cual muestra gran similitud en los objetivos que persigue.

2.3.2.1 YCB

El dataset Yale-CMU-Berkeley (YCB) Object and Model Set Calli y cols. (2015) será la base para nuestro dataset de agarres, y consiste en una serie de objetos de uso diario con diferentes tamaños, texturas, formas, pesos y rigidez. Este dataset ha sido creado para facilitar la evaluación de la manipulación robótica. El dataset está formado por los objetos reales, y sus correspondientes modelos 3D escaneados a alta resolución. Los objetos reales pueden ser adquiridos para contar con ellos en la vida real, siendo muy útiles para la validación sobre el mundo real de sistemas entrenados con datos sintéticos. Incluso sin contar con los objetos reales, el dataset puede resultar de mucha utilidad dado que existen datasets derivados de estos modelos, como BOP (Hodaň y cols., 2018), que está compuesto por secuencias de vídeo de los objetos reales del dataset YCB. En nuestro caso, es interesante la posibilidad que nos ofrece este dataset a la hora de crear agarres humanos o de robótica. Tanto es así, que es altamente reconocido y empleado en proyectos relacionados con análisis y predicción de agarres, modelado de poses de la mano, etc.

En la Figura 2.3 se pueden observar diferentes objetos con el fin de ejemplificar cómo van a ser los modelos a partir de los cuales se van a realizar los agarres. Es interesante que sean de un tamaño tal que puedan agarrarse con una única mano.

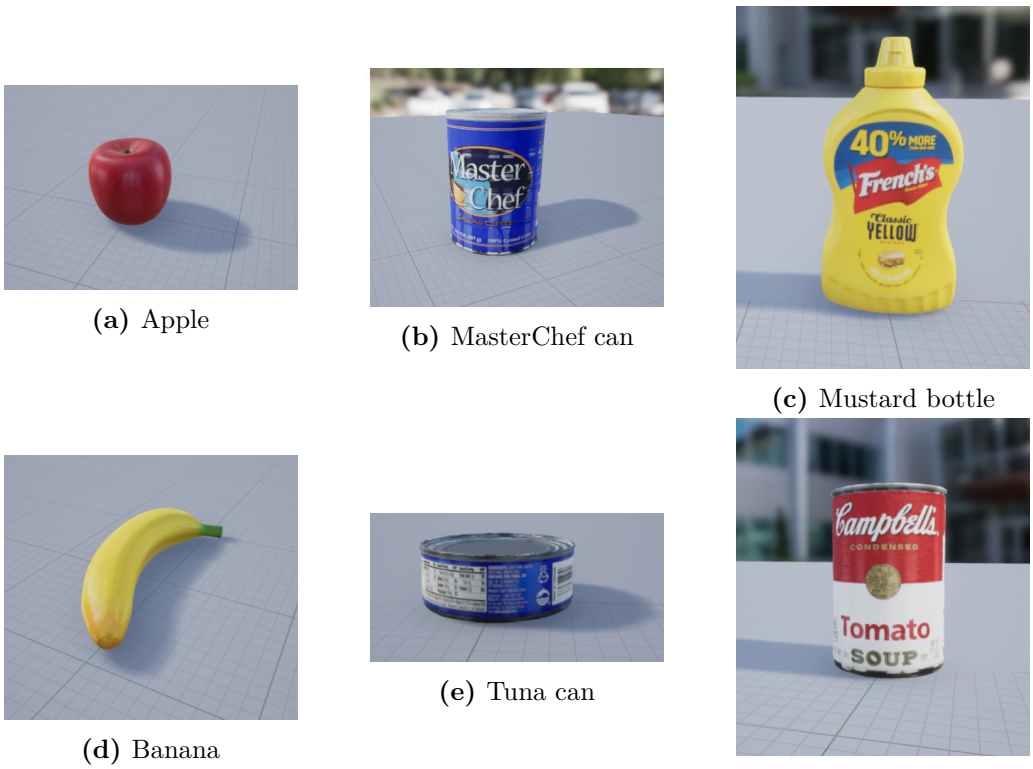
2.3.2.2 Dex-YCB

Dataset enfocado a estimación de poses de la mano (Chao y cols., 2021). Para esto han creado un sistema con 8 cámaras con las que capturar la información de agarres reales de objetos en una mesa desde diferentes vistas de forma síncrona. Este dataset, por tanto, está formado por 582K frames RGB-D con 1000 secuencias de 10 personas diferentes agarrando 20 objetos desde las 8 cámaras que se disponen, las cuales capturan datos simultáneamente a 30 Frames Per Second (FPS) a color y con una resolución de 640 x 480. En este proyecto se divide la mano en 21 articulaciones, 4 por dedo más la muñeca.

Finalmente, en este proyecto tratan de evaluar una nueva tarea relacionada con la robótica colaborativa que consiste en la generación de agarres seguros de objetos entre humanos y robots. Es decir, dado un objeto sostenido por un humano, trata de encontrar el mejor agarre del robot sin poner el peligro a la persona.

2.3.2.3 ContactDB

ContactDB (Brahmbhatt, Ham, y cols., 2019) extrae los contactos de agarres de manos humanas sobre objetos a través de cámaras térmicas. Esto es posible debido a que cuando la mano se pone en contacto con el objeto le traslada calor a través del área de unión. En su trabajo se presentan tres contribuciones principales:



(a) Apple

(b) MasterChef can

(c) Mustard bottle

(d) Banana

(e) Tuna can

(f) Tomato soup

Figura 2.3: Ejemplos del dataset YCB

- **Dataset** de mapas de contacto que captura la calidad del *grasping* humano para 3750 objetos principalmente del hogar, junto con más de 375K datos térmicos RGB-D.

- **Análisis**: estudio de la influencia de la forma del objeto, así como del tamaño y lo que se desea hacer con el objeto.

- **Predicción**: investigación de representaciones de datos y algoritmos de predicción para predecir el mapa de contacto a partir de la forma de un objeto.

En cuanto a la metodología empleada y la selección de objetos con los que se han hecho las pruebas iniciales, se han preparado 50 objetos, entre ellos algunos del dataset YCB, típicamente del hogar, asegurándose de que tienen formas diferentes, un tamaño adecuado para ser agarrados y transportados correctamente, además de que no sean deformables. Durante las pruebas se ha llegado a la conclusión de que se obtienen mejores resultados imprimiendo en 3D estos objetos que empleando directamente el objeto real. Esto es debido a que el material PLA utilizado para impresión 3D retiene bien la temperatura, además de que su color homogéneo ayuda a otras tareas como la segmentación de los objetos.

Los objetos sobre los que se han realizado los agarres en pruebas reales son procesados utilizando una cámara térmica montada junto a una Kinect RGB-D, las cuales fueron calibradas con ROS, para ser capaces de obtener la imagen térmica con una gran precisión.

La forma en la que se adquirieron los datos es la siguiente: Se preparan 50 objetos impresos en 3D, una persona con guantes especiales para transferencia de calor se encarga de cogerlos, mantenerlos durante 5 segundos, y finalmente entregarlo al experimentador el cual lleva un guante especial que aísla la transferencia de calor. El experimentador se encarga de dejarlo en una mesa giratoria, la cual rota 360 grados para obtener 9 frames desde diferentes perspectivas del objeto mediante las cámaras, y así obtener datos de profundidad y térmicos en los que observar donde ha habido contacto.

Tras realizar el procesamiento de datos, se genera un mapa de contactos con la información deseada.

Se hicieron pruebas tanto usando un objeto como entregándolo al objetivo, obteniéndose los mapas de calor de la Figura 2.4. Como es lógico, si nos fijamos en los prismáticos por ejemplo, al utilizarlos obtendrán calor de ambas manos, sin embargo si vamos a transportarlo, solo los cogemos con una mano.

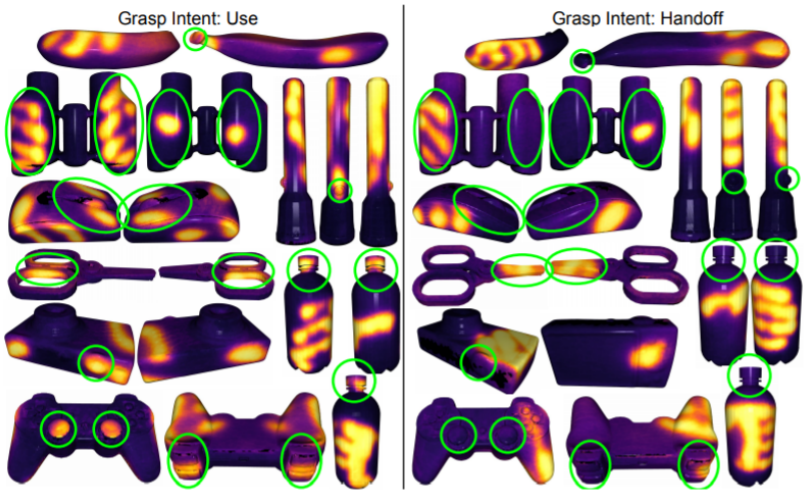


Figura 2.4: Mapas de calor ContactDB(Mahler y cols., 2019)

Para llevar a cabo el **análisis** de los datos obtenidos se llevan a cabo diferentes estudios que analizan los tipos de grasping dependiendo de las cualidades del objeto. Una de las conclusiones que obtienen con el análisis es la motivación de la inclusión de zonas que no son la punta del dedo en los algoritmos de predicción del agarre, ya que hay objetos que son agarrados con más partes de la mano, siendo una gran oportunidad y/o avance a la hora de EOAT robóticos blandos.

El último proceso llevado a cabo en este proyecto está relacionado con la **predicción** de agarres, se lleva a cabo experimentación para predecir mapas de contacto basados en la forma de los objetos. Para esto investigan dos tipos representaciones del objeto, vista única RGB-D y 3D completa.

2.3.2.4 ContactGrasp

ContactGrasp (Brahmbhatt, Handa, y cols., 2019) es un framework cuyo objetivo es la síntesis funcional de agarres a partir de la forma y el contacto del objeto. Hace uso del

dataset comentado previamente en la sección 2.3.2.3, ContactDB. Y plantea una nueva forma de procesar el mapa de calor extraído de los objetos.

Las dos contribuciones que que describe en su paper son las siguientes:

- Desarrollo de una representación de contacto multipunto que soporte una síntesis eficiente del agarre.
- Proposición de una aproximación *sample-and-rank* (muestra y rango) para una síntesis de agarre funcional a partir de la forma del objeto y el contacto en la superficie del objeto, que puede trabajar con múltiples modelos de mano.

Se ha demostrado la efectividad de este framework sintetizando agarres funcionales para 3 tipos de mano diferentes (Figura 2.5), 19 objetos y 2 intentos funcionales.

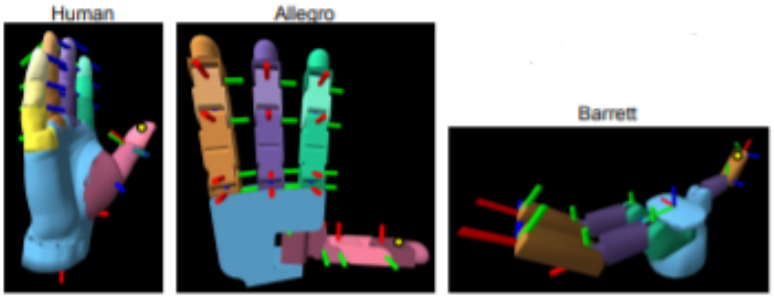


Figura 2.5: Manos ContactGrasp (Brahmbhatt, Handa, y cols., 2019)

2.3.2.5 Dexterity Network

También llamado Dex-net (Mahler y cols., 2019), es un proyecto de investigación creado por el profesor Ken Goldberg y Jeff Mahler en la Universidad de California que consiste en la generación de conjuntos de datos para entrenar una Grasp Quality Convolutional Neural Networks (GQ-CNN), que predicen la probabilidad de éxito en los agarres de diferentes objetos a partir de nubes de puntos.

De esta forma, Dex-Net proporciona código, datasets y algoritmos para generar los datos necesarios para calcular la robustez del grasping, y tiene como objetivo encontrar el mejor agarre del robot para más de diez mil modelos 3D.

Un dato de interés de Dex-Net, es la Mean Picks Per Hour (MPPH) que consigue, cuyo resultado se puede obtener con la ecuación 2.1. Siendo *TMpick* el tiempo medio por pick, y *PMexito* la probabilidad media de éxito.

$$MPPH = TMpick * PMexito \tag{2.1}$$

Los humanos son capaces de hacer entre 400 y 600 MPPH, mientras que en una competición organizada por *Amazon* hace unos años los mejores robots estaban entre 70 y 95 MPPH. Sin embargo, Dex-Net ha conseguido entre 200 y 300 MPPH lo que supone una mejora a tener muy en cuenta.

3 Metodología

La metodología empleada en este proyecto se divide en dos partes: hardware y software. En el apartado hardware se explicarán los elementos físicos empleados, mientras que el software lo conforman el conjunto de programas con los que se ha llevado a cabo el entorno, configuración y resto del trabajo.

3.1 Hardware

En este apartado se van a discutir los motivos de la elección de las gafas de realidad virtual empleadas, además de describir sus características y las del ordenador en el que se ha trabajado.

3.1.1 Oculus Quest

La elección de las gafas de realidad virtual es uno de los puntos de partida. Teniendo presente que necesitamos realizar un tracking de manos se seelecciona a la tecnología creada por Oculus en 2019, la cual utiliza los propios sensores de las gafas Oculus Quest para este *hand tracking*, es decir, es capaz de obtener la posición y movimiento de las manos en tiempo real haciendo uso de las 4 cámaras integradas. A esta necesidad se le suma que dentro de los dispositivos, de los que dispone el departamento, se encontraban estas gafas por lo que la elección fue clara ya que cumplen lo que buscamos y están a nuestra disposición.

Las Quest están formadas por las gafas de realidad virtual que integran 4 cámaras y el audio, y dos mandos ergonómicos. El sistema de las cuatro cámaras integradas en las Oculus Quest fue algo novedoso en su salida, ya que daba un salto en la libertad del sistema con respecto a las predecesoras que necesitaban hardware externo para traducir nuestra posición en el mundo real al virtual. Esto fue bautizado como *Oculus insight*.

Estas gafas también han sido llamadas *Oculus Quest All-in-one*, y reciben este nombre ya que, además del sistema de cámaras comentado previamente, integra la mayoría de las funcionalidades de las Oculus previas, siendo capaces de utilizarse sin cableado, ni para alimentación ni para comunicarse con el PC, aunque ofrece la posibilidad de su uso mientras se alimentan. A continuación se muestran sus características:

Panel y resolución	OLED 1600x1440px por ojo
SO	Basado en Android
Movimiento	6 Grados de libertad
Velocidad de refresco	72Hz
Conector	USB Type-C
CPU	Qualcomm® Snapdragon 835
GPU	Qualcomm® Adreno™ 540
Memoria RAM	4 GB

Figura 3.1: Gafas Oculus Quest y controladores

3.2 Robot Toyota HSR

Toyota está tratando de evolucionar hacia un planteamiento global de movilidad, como bien declaró el presidente Akio Toyoda en la CES (Consumer Electronics Show) expo en 2018: *"It's my goal to transition Toyota from an automobile company to a mobility company, and the possibilities of what we can build, in my mind, are endless."*. Es por esto que cada vez están tratando de llevar más allá su tecnología, habiendo entrado de lleno en el campo de la robótica, más en concreto con este robot Human Support Robot (HSR)(Upton, s.f.) a la robótica colaborativa. Las especificaciones técnicas de este robot son:

- Diámetro del cuerpo: 430mm
- Altura del cuerpo: 1005-1350mm
- Peso: 37kg (approx.)
- Altura de los hombros: 340-1030mm
- Capacidad de sujetar objetos: peso hasta 1.2kg; ancho hasta 130mm
- Velocidad máxima: 0.8 km/h
- Tiempo medio de ejecución: 2-3 horas
- Software:
  - JetPack, L4T
  - Robotic Operating System (ROS)
  - OpenCV
  - CUDA
- Sensores/componentes:

- Micrófonos
- Cámara RGB-D
- Cámara gran angular
- Sensor de par de fuerza
- Cámara estéreo
- IMU
- Sensor láser de distancia.

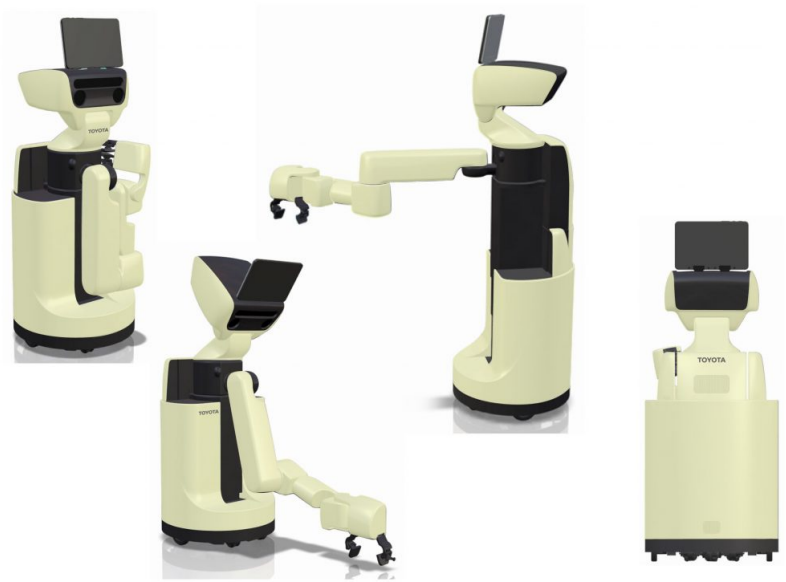


Figura 3.2: Imágenes robot HSR.

Este robot está diseñado para asistir al humano en el hogar y puede ser manejado por voz o a través de una tablet. Tiene un cuerpo cilíndrico altamente maniobrable, es capaz de interactuar con el entorno ya sea esquivando obstáculos o cogiendo objetos haciendo uso de su brazo plegable y una pinza robótica, tal y como se puede ver en la Figura 3.2.

Este robot permite la teleoperación con el rostro y la voz del operador transmitidos en tiempo real para simular una total interacción humana. Esta tarea se prevé que será altamente demanda, dado el envejecimiento de la sociedad, ya que da soporte a una ayuda real en el día a día.

3.2.1 Ordenadores empleados

El ordenador portátil personal sobre el que se han realizado diferentes pruebas consta de las siguientes características:

- Procesador i7-7700HQ
- 8 GB RAM

- Nvidia GTX 1050 2Gb

Además, el sistema operativo empleado para ejecutar el software junto con las gafas Oculus Quest ha sido Windows 10. Como se puede observar, se trata de un ordenador modesto el cual será capaz de ejecutar el entorno con unas especificaciones bastante básicas. Es por esto que además se empleará el ordenador del laboratorio para hacer pruebas de una forma más fluida. Sus características son las siguientes:

- Procesador i3-8100
- 8 GB RAM
- Nvidia RTX 2080 8Gb

El dataset ha sido conformado en el ordenador del laboratorio ya que dispone de una tarjeta gráfica de gama alta capaz de ejecutar con una mayor fluidez el motor gráfico y el entorno preparado con los objetos del dataset YCB, lo que proporciona una gran comodidad al usuario.

3.3 Software

Para realizar los objetivos marcados se van a instalar los programas necesarios, tanto para el uso de las gafas de realidad virtual de la empresa Oculus, como el software en el que se programará el entorno de trabajo.

3.3.1 Oculus

Para hacer uso de las gafas en el ordenador tenemos que emparejarlas mediante su software oficial, el cual se obtiene desde su propia página[1]. Para esto se debe conectar el extremo del cable USB-C a las gafas y a continuación el otro extremo del cable al un puerto USB 3.0 del equipo y finalmente añadir el dispositivo en el programa.

Una vez vez se ha acabado el proceso está todo preparado, y solo queda iniciar las gafas Oculus con el software corriendo en el ordenador y asegurarse de que la opción *Oculus Link* está activada. Esta funcionalidad permite utilizar las Oculus Quest, un dispositivo autónomo que no necesita un ordenador para generar las imágenes que muestra, como si fueran unas Oculus Rift, es decir, para visualizar aquello que es procesado en el ordenador. Esto es crucial para poder utilizar las Oculus Quest mientras desarrollamos un proyecto de realidad virtual en Unreal Engine.

Los requisitos mínimos de este software son los siguientes:

Componente	**Requisitos mínimos**
Procesador	Intel i3-6100/AMD Ryzen 3 1200, FX4350 o superior
Tarjeta gráfica	NVIDIA GTX 1050 Ti/AMD Radeon RX 470 o superior
Tarjeta gráfica alternativa	NVIDIA GTX 960 4GB/AMD Radeon R9 290 o superior
Memoria	8 GB de RAM o más
Sistema operativo	Windows 10
Puertos USB	1 puerto USB 3.0
Salida de vídeo	Salida de vídeo miniDisplayPort compatible

[1] `https://www.oculus.com/setup/`

En cuanto a estos, en un principio mi ordenador portátil podría haber dado algún problema ya que la gráfica dedicada recomendada es NVIDIA GTX 1050 Ti, mientras que la de mi ordenador personal es una NVIDIA GTX 1050. Pese a esto hemos podido iniciar el software Oculus sin problemas probablemente debido a que cumple los requisitos mínimos de la tarjeta gráfica alternativa.

Y los requisitos recomendados se muestran a continuación:

Componente	**Requisitos recomendados**
Procesador	Intel i5-4590/AMD Ryzen 5 1500X o superior
Tarjeta gráfica	NVIDIA GTX 1060/AMD Radeon RX 480 o superior
Tarjeta gráfica alternativa	NVIDIA GTX 970/AMD Radeon R9 290 o superior
Memoria	8 GB de RAM o más
Sistema operativo	Windows 10
Puertos USB	1 puerto USB 3.0
Salida de vídeo	Salida de vídeo DisplayPort compatible

3.3.1.1 Características e implementaciones

Con el paso del tiempo, Oculus trata de innovar y lanzar nuevas implementaciones. Vamos a comentar las dos más interesantes sobretodo en vistas de este TFM.

La tecnología de realidad virtual que ofrece Oculus ofrece un alto grado de inmersión, por lo que puede implicar algún tipo de peligro al interactuar con el entorno que nos rodea en el mundo real, surge así el reto de proporcionar algún tipo de ayuda para permanecer seguros, en otro caso el usuario tendrá que pausar a menudo lo que esté haciendo y quitarse las gafas para cualquier tipo de interacción con el mundo real.

En este sentido, Oculus ideó una solución completa a este problema, basada en la reconstrucción en tiempo real y la síntesis de la vista estereoscópica del entorno del mundo real, algo que se apoda como **passthrough** (Chaurasia y cols., 2020) y consiste básicamente en aprovechar las cámaras estereoscópicas de las que disponen las gafas para proporcionar una imagen en tiempo real del entorno que nos rodea. Las imágenes que proporciona son en la dirección en la que las gafas apuntan, y además son tratadas para ser mostradas al usuario de manera coherente a la posición de sus ojos, creando la sensación de ver a través de las gafas.

El *passthrough* se activa solo cuando sales de la zona segura definida inicialmente en el sistema guardián para garantizar tu seguridad. Además, puedes activarlo dando tres toques en el lateral derecho de las gafas.

Por otra parte, generalmente las tecnologías de gafas de realidad virtual llevan consigo una serie de periféricos para interactuar en los entornos de RV y RA, como pueden ser mandos o incluso guantes sensorizados. Desde Oculus surge la idea de eliminar la necesidad de estos periféricos realizando un seguimiento de las manos en tiempo real haciendo uso, de nuevo, de las cuatro cámaras de las que disponen las gafas. Este **handtracking** (Han y cols., 2020) debe ser lo suficientemente robusto como para producir la estimación de la pose de la mano en diferentes entornos y con cualquier mano.

Para la detección de manos, el primer objetivo que se presenta es encontrar el *bounding box* donde se encuentran las manos. Para esto se crea un dataset específico y se crea una Convolutional Neural Network (CNN) a la cual se ha llamado DetNet.

Puesto que pasar las 4 imágenes de las 4 cámaras por la red DetNet para cada uno de los frames es una tarea muy costosa computacionalmente, se implementa una aproximación cuando la mano a seguir esta disponible, es decir, cuando se ha localizado en algún frame. Primero se extrapola la postura actual de la mano a partir de las dos poses anteriores de forma que:

$$\hat{\theta}_t = 2\theta_t - 1 - \theta_t - 2$$

Este handtracking aprovecha la pose extrapolada θ_t proyectando los puntos clave de la mano en cada cámara y calculando el círculo mínimo que la engloba como resultado de la detección. En el caso de que no se encuentre la mano actual, se ejecuta DetNet para el siguiente fotograma.

3.3.2 Unreal Engine 4

El programa utilizado para crear el entorno virtual y extraer los datos etiquetados es Unreal Engine 4 (UE4), concretamente la versión 4.26.2. Es la base de este TFM ya que ejecutaremos todo lo necesario aquí, lo cual nos hará de puente entre las gafas y el resto del proyecto.

Figura 3.3: Logo Unreal Engine

Este software es un entorno de desarrollo creado por a empresa Epic Games basado en código C++ y creado en 1995, aunque no fue hasta 2015 cuando se lanzó su primera versión libre y gratuita. Con los años ha ido ganando fama ya que, pese a que inicialmente se crea como un motor de juegos, cada vez su uso es más amplio en entornos más que interesantes como pueden ser el cine, simuladores, etc.

Este motor incluye una gran cantidad de herramientas con las que crear el entorno deseado. En nuestro caso el motivo principal de su elección, además de que es uno de los motores de videojuegos más potentes del mercado, es su carácter gratuito. Nos permite integrar el framework necesario para trabajar con las gafas Oculus de una forma sencilla, programando o bien con C++ o con Blueprints, que es una forma de programación mediante scripting visual característica de Unreal Engine.

4 Desarrollo

En este capítulo se van a describir los diferentes estudios e implementaciones realizadas en este proyecto. Para ello se tratarán de exponer claramente los objetivos de cada apartado y las acciones desarrolladas para cumplirlo. Primero trataremos la base empleada en nuestro proyecto comentando su funcionamiento, posteriormente describiremos el entorno empleado donde aparecen los objetos y la construcción del dataset donde se hará hincapié en la metodología de guardado y etiquetado. Una vez introducido el entorno y etiquetado procedemos a explicar el desarrollo que se ha llevado a cabo para diferenciar cada uno de los dedos junto con el script realizado para la obtención del mapa de calor.

4.1 Desarrollo base

Este trabajo utiliza como base el proyecto *unrealhandgrasp* (Mulero Pérez, 2021), que implementa el *handtracking* de Oculus para hacer el seguimiento de las manos y además desarrolla la lógica del sistema de agarre que se va a emplear. La lógica de agarre de la que parte consideraba que un objeto se agarraba si el pulgar está en contacto con el objeto o el objeto se encuentra dentro de la esfera de colisión y además colisiona el dedo índice. Tras su desarrollo, es posible realizar agarres con dos dedos, siempre que al menos uno de ellos sea el dedo pulgar o índice.

Para la lógica de agarre se llevan a cabo diferentes configuraciones para calcular el contacto de los dedos y la palma, y el nivel de apertura de la mano. Esto ayuda a decidir si el objeto esta agarrado o no.

Cada dedo tiene una cápsula unida a su falange distal y media, la cual es componente propio de Unreal Engine que se utiliza para saber si un objeto la ha traspasado. Además, en la palma de la mano habrá un componente esférico cuyo comportamiento es el mismo que la cápsula de los dedos, para saber si el objeto puede ser agarrado se comprobará esta esfera de colisión, si el objeto la ha traspasado puede ser agarrado, en cambio, si el objeto sale del rango de la esfera su agarre deja de ser posible.

Con las cápsulas y la esfera de colisión ya sabemos si la mano está en contacto con el objeto y tenemos la información de cada uno de los dedos, el siguiente paso para la lógica del agarre es bloquear los huesos del dedo que está en contacto con el objeto, ya que no debe cerrarse más aunque el handtracking de Oculus calcule que la mano está más cerrada.

Además, se impone la condición de que la mano tiene que estar lo suficientemente cerrada para considerarse agarre. Para esto se calcula la distancia de cada una de las yemas de los dedos a la muñeca y se comprueba que esté por debajo de un umbral el cual se ha determinado empíricamente. Una vez se ha considerado que la mano ha agarrado un objeto se va a ir comprobando la apertura de la mano en comparación con la medida guardada cuando se ha considerado agarre, si es menor o igual se mantiene el agarre ya que la mano no se ha abierto lo suficiente como para considerarse que el objeto se ha soltado.

De esta forma para agarrar el objeto se recomienda cerrar la mano por completo y en Unreal Engine se bloquearán los dedos en las posiciones donde han contactado con el objeto, y para soltarlo abrirla y así dejando de estar en contacto por completo para soltar el objeto y seguir el *handtracking* normalmente.

El entorno del proyecto previo comentado anteriormente estaba formado por diferentes puestos o zonas, cada uno de ellos conteniendo objetos agarrables de temáticas diferentes como son: ingeniería, juegos, medicina, química-física, arquitectura y gastronomía. Puesto que en nuestro trabajo no necesitamos separar dependiendo del ámbito, sino que usaremos los objetos del dataset YCB, no se va a hacer uso de este entorno.

Una de las características más importantes en la interacción humano-entorno es la forma de moverse del usuario, ya que ahora se debe tratar de emplear las manos y el *handtracking* en lugar de los mandos de Oculus. Para esto se crea un gesto que permite al usuario moverse hacia delante en la dirección en la que mira, se puede observar fácilmente en la Figura 4.1. La velocidad de movimiento se puede regular con la pose de la mano, y viene dada por lo estirados que estén los dedos índices, lo cual es calculado mediante la distancia entre el dedo índice y pulgar. Como requisito antes de moverse a través del entorno es que no se este agarrando ningún objeto, ya que podría dar lugar a confusión un tipo de agarre con posiciones de los dedos pulgar e índice semejantes a las requeridas para movimiento.

Figura 4.1: Vista del usuario moviéndose por el entorno. (Mulero Pérez, 2021)

4.2 Entorno

La creación del entorno es uno de los pasos básicos en este TFM. Se tiene que facilitar al usuario la movilidad suficiente para ser capaz de ir hasta donde los objetos a agarrar van apareciendo, y posicionarse para realizar un agarre cómodo y real. Para esto, el entorno tiene que ser lo más sencillo posible para que pueda ser utilizado por cualquier usuario sin necesitar tiempo excesivo de adaptación/aprendizaje.

En nuestro caso no necesitaremos las diferentes zonas separadas por temática puesto que

nos centramos exclusivamente en objetos del dataset YCB. Con el objetivo de facilitar las pruebas con cualquier persona, sin importar su experiencia previa, los objetos irán cayendo de uno en uno sobre una mesa a la cual se puede acceder en apenas segundos trasladándose hacia ella.

4.3 Construcción del dataset

La primera parte de este trabajo consiste en la creación de un dataset de agarres sobre todos los objetos del dataset YCB.

Para esto se hace uso de un entorno virtual realizado en UE4, el cual ha sido explicado en la Sección 4.2. Una vez tenemos el objeto, procederemos a cogerlo con nuestra mano. El programa haciendo uso del handtracking incorporado en el framework de la compañía Oculus seguirá los movimientos de las manos en tiempo real. El funcionamiento será el siguiente: se dispone de dos manos virtuales creadas en UE4 en las cuales se vuelca la información del handtracking de las gafas, para coger un objeto cerramos la mano de forma que en UE4 se cierra hasta el espacio virtual del objeto, el cual podrá ser sostenido observándose todos los puntos de contacto.

El objetivo es realizar un número estipulado de agarres que a priori consideramos exitosos sobre un mismo objeto, guardar los puntos de contacto de la mano con el objeto junto con la malla y la textura original, y finalmente juntar todos los agarres para formar un mapa de calor con todas las zonas de contacto. Además, este procedimiento será realizado con ambas manos y por más de una persona a modo de hacerlo lo más genérico y cercano a la realidad posible.

4.3.1 Etiquetado de datos

Para cumplir este objetivo se han elegido una serie de teclas para facilitarnos la automatización y velocidad a la hora de crear el dataset:

- **Espacio**: Pasa al siguiente objeto del dataset de YCB.
- **J**: Reaparece el mismo objeto que esta en la escena.
- **T**: Guarda la textura.
- **G**: Limpia la textura.

La metodología a seguir a la hora de guardar cada agarre será la siguiente: una vez aparece automáticamente el primer objeto en el entorno y cae en la mesa, agarrarlo correctamente y pulsar T para guardar la textura. Si el agarre no ha sido exitoso se debe pulsar G para limpiar la textura e intentarlo de nuevo, teniendo la posibilidad de hacer que el objeto reaparezca con J si consideramos que la posición no nos deja coger el objeto cómodamente. Una vez se ha guardado exitosamente se procede a pasar al siguiente objeto con la tecla *Espacio*. La lógica hecha en UE4 esta hecha de forma que diferencia la mano con la que se ha realizado el agarre y guarda la textura teniéndolo en cuenta.

La forma en la que se guarda tiene relación con el objeto y el momento en el que se hace el agarre, de forma que para el objeto número 14 del dataset YCB apodado *014_lemon* se

crearía una ruta de carpetas dentro de HeatMaps\SingleFingers con una subcarpeta llamada como el objeto, y dentro, otra subcarpeta para cada agarre exitoso cuyo nombre viene dado por el momento en el que se ha pulsado la tecla que da la orden de guardar. Un claro ejemplo puede ser la ruta HeatMaps\SingleFingers \014_lemon\0623120837. Esta ruta ha sido creada para guardar un agarre del objeto 14 y la información de los números es la siguiente:

- 06: mes
- 23: día del mes
- 12: hora
- 08: minuto
- 37: segundos

Aunque en la carpeta de Google Drive que se adjunta en la Sección 5, cada agarre se diferencia mediante un número a diferencia de lo creado con UE4 a modo de hacerlo más visible y sencillo de emplear en proyectos futuros.

Llegando a la precisión de segundos se consigue evitar que varios agarres se guarden en una misma carpeta. Además, cada contacto individual se guardará con la información del objeto sumándole el dedo que es y un índice que indica el número de agarres guardados para este mismo objeto.

La información que dispondrá nuestro dataset creado con todos los objetos de YCB será:

- Una textura por agarre y dedo con los contactos con las coordenadas UV.
- Pose de la mano en el momento del agarre, dada por la posición y rotación de cada una de las articulaciones de la mano.
- Posición y rotación del objeto.

Toda la información relacionada con posición y rotación viene dado en coordenadas del mundo.

Este dataset[1], una vez completado, puede ser muy valioso para tareas posteriores como: estudio en profundidad de los agarres, entrenamiento de Inteligencia Artificial capaz de predecir la mejor pose de la mano para objetos diferentes a partir de la experiencia obtenida con estos datos estructurados o bien capaz de evaluar el índice de bondad del agarre.

4.4 Obtención de los puntos de agarre como mapas de calor

Para obtener los puntos del objeto sobre los que se ha establecido un contacto que ha permitido un agarre exitoso, necesitamos recuperar esa información de contacto de alguna forma en UE4. Al entrar en contacto con otro objeto, las cápsulas que dispusimos sobre los dedos para detectar colisiones y adaptar la pose de la mano al objeto agarrado nos facilitan la información del punto de contacto, que podemos convertir a las coordenadas que necesitemos. No obstante, esta no será la estrategia que utilizaremos para obtener esta información. En

[1] https://github.com/3dperceptionlab/unrealhandgrasp/tree/dataset

su lugar, dibujaremos en una textura un mapa de calor que resaltará aquellos puntos sobre los que se ha ejercido contacto. Esta textura será dibujada en tiempo real a través de un material de Unreal Engine, que funciona como un *shader*. En este caso, le inyectamos a este material/*shader* las coordenadas UV[2] del punto sobre la superficie del objeto sobre el que se ha realizado el contacto, y pintamos de blanco dicho punto, y dispersamos hacia el negro con un radio determinado, de tal forma que obtengamos un área de influencia en lugar de un punto único. Estos materiales se pueden configurar como aditivos, de tal forma que estas modificaciones se vayan acumulando, generando un mapa de calor con las zonas sobre las que la mano hace el contacto con el objeto. Este mapa de calor es una imagen que está codificada de la misma forma que la textura que se aplica sobre la malla 3D del objeto, por lo que al aplicarla sobre el mismo, obtendremos el volumen con los puntos de contacto. También se puede combinar con la textura original para resaltar los puntos de agarre manteniendo el color original del objeto. En la figura 4.2 se puede apreciar la implementación de este material.

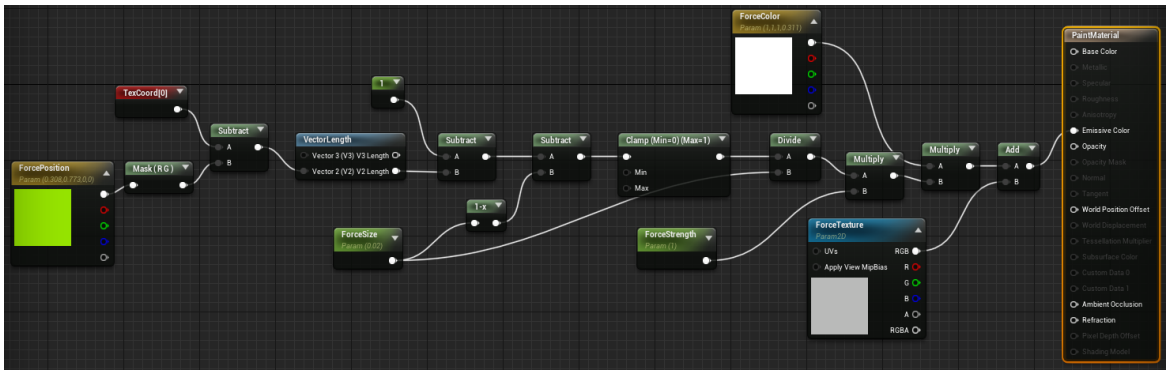
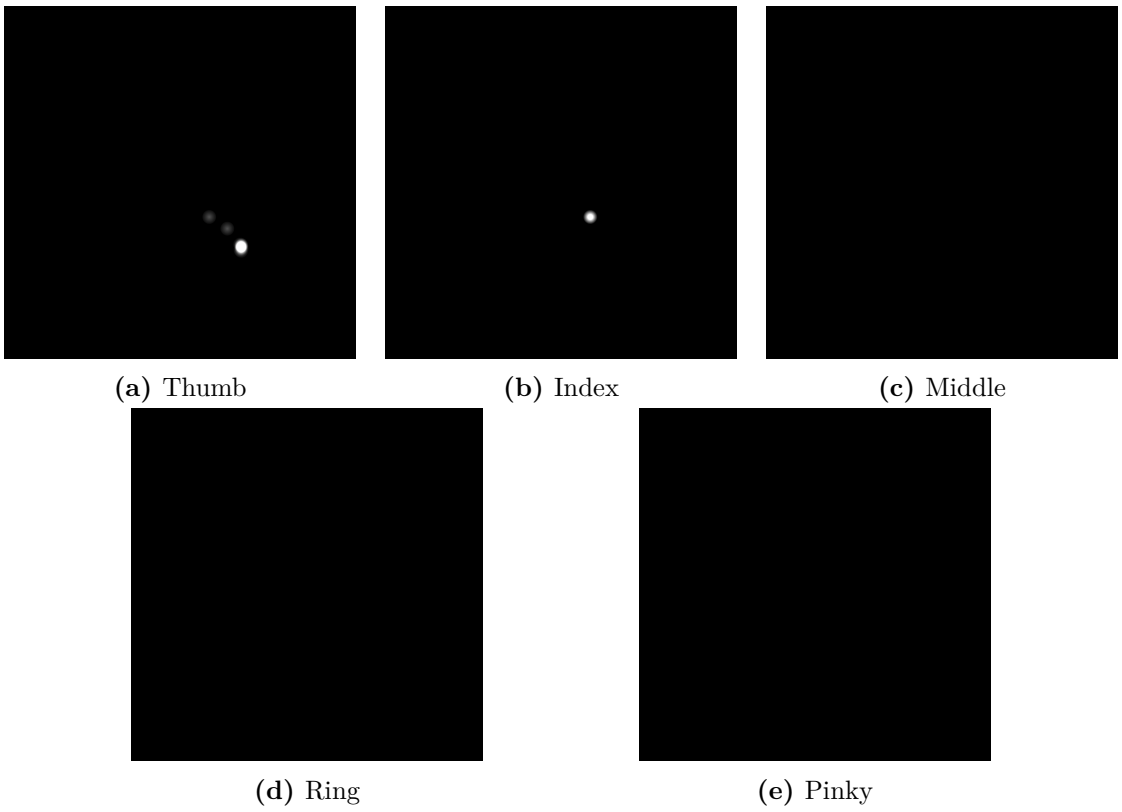


Figura 4.2: Material que implementa la generación de los mapas de calor dados los puntos de contacto. Las coordenadas UV se le inyectan al shader en *ForcePosition* codificadas como un dato RGBA (en R y G). El shader colorea más o menos blanca la textura en función de la distancia lineal (más o menos cerca) al punto de contacto.

4.4.1 Agarres individuales o por dedos

Para contar con toda la información posible sobre los agarres, vamos a obtener una textura independiente con los puntos de contacto para cada dedo. Para ello tendremos que tener 5 materiales dibujando la textura en paralelo, cada uno de ellos atendiendo al contacto realizado por cada una de las cápsulas en cada dedo. Al contar con esta información por separado, podremos aprovecharla mejor al poder usarla de manera independiente, o agregarla en una sola textura.

[2]Las coordenadas UV se refieren al mapeo de una texturas sobre una malla 3D. La textura es una imagen cuadrada cuyos píxeles, identificados con una coordenada U y otra V, se corresponden con un punto de la superficie de la malla. También se puede ver como los polígonos que componen la malla se despliegan en 2D sobre la textura. En Unreal contamos con un método que nos devuelve la coordenada UV sobre la textura dado un punto de la superficie de una malla.



(a) Thumb **(b)** Index **(c)** Middle

(d) Ring **(e)** Pinky

Figura 4.3: Dedos agarre *Lemon*

4.4.2 Mapas de calor

Tras haber logrado obtener el contacto de cada dedo individualmente, vamos a procesarlos para poder ver y diferenciar cada uno de ellos.

De UE4 obtenemos los puntos de contacto en escala de grises y en formato High Dynamic Range (HDR), por lo que tenemos que trabajar con ellos para obtener un mapa de calor único donde se puedan observar y estudiar posteriormente.

Para ello se va a crear un script de Python en el que se van a llevar a cabo una serie de pasos:

1. Abrir y procesar cada uno de los archivos donde se encuentran los puntos de contacto de los dedos.

2. Creación de una textura que conforme el mapa de calor agregado que replica una cámara térmica.

3. Creación de una textura que conforme el mapa de contacto agregados con los dedos diferenciados por color.

4. Aplicación de las tres texturas (original, mapa de calor y mapa de contactos con colores por dedo) a la malla 3D y obtención de imágenes desde varios puntos de vista.

Puesto que tenemos un fichero para cada dedo, inicialmente indicamos la ruta donde se encontrarán y comprobaremos que se han abierto correctamente para poder seguir avanzando. Una vez los tenemos abiertos comenzamos el algoritmo que dará lugar al mapa de calor. Este mapa de calor será una textura del mismo tamaño que los ficheros individuales pero donde se va a diferenciar cada uno de los dedos además de la intensidad en cada punto. En los ficheros originales obtenemos los contactos en escala de grises, lo que quiere decir que donde sea negro no hay contacto y donde sea blanco hay un contacto completo, además se diferencia el nivel de intensidad en el contacto con esta escala de grises ya que hace uso de valores intermedios.

Para obtener el mapa de calor, se ha establecido un degradado lineal que interpola entre un morado oscuro (que representa zonas frías), un rojo anaranjado (que representa zonas templadas) y un amarillo (que representa zonas calientes), en función del valor de intensidad agregado obtenido de sumar las contribuciones de los cinco dedos.

Por otro lado, para lograr obtener los mapas de contacto con los dedos diferenciados por color, usará el mismo valor de intensidad, pero aportando cada dedo únicamente a un color determinado. Para esto lo más sencillo y vistoso ha sido emplear los colores primarios RGB (azul, verde y rojo), apodados así porque no se pueden obtener de ningún otro color. Sin embargo, como hay solo tres canales de color en la imagen (RGB) y cinco dedos en total, se utilizarán dos de los tres secundarios (amarillo, magenta). Por tanto, los colores que se van a asignar a cada dedo son los siguientes. También se especifica su valor cuando la intensidad es máxima. El valor no es 255 para evitar que si existen solapamientos entre varios dedos se sature rápidamente:

- Pulgar(*Thumb*): Verde (G 127)
- Índice(*Index*): Rojo (R 127)
- Medio(*Middle*): Azul (B 127)
- Meñique(*Pinky*): Amarillo (R 63, G 63)
- Anular(*Ring*): Magenta (R 63, B 63)

La metodología final consiste en recorrer las texturas de los contactos y volcarlas en nuestra nueva textura con el color que le corresponda, y finalmente guardarla en un fichero .png a modo de facilitar su visionado con cualquier software de imagen.

Una de las salidas principales de este proceso se puede observar en la Figura 4.4, donde se crean los mapas de calor de los cinco contactos que se muestran en la Figura 4.3. En la siguiente Sección 5 se tratarán de mostrar más objetos junto con su textura y todas las salidas que se han llevado a cabo.

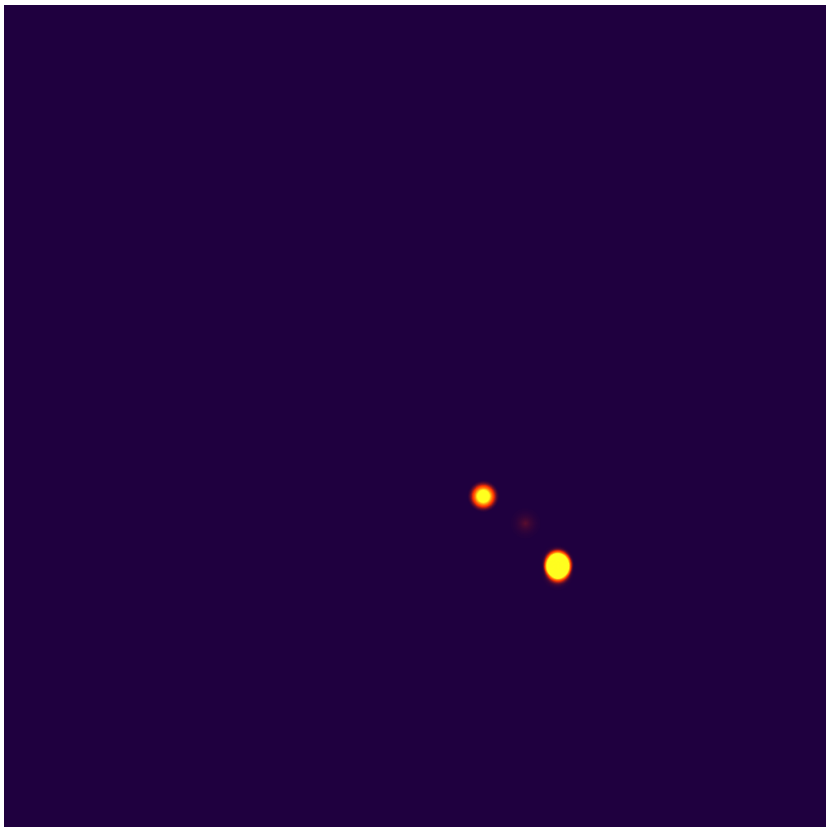


Figura 4.4: Mapa de calor de los contactos mostrados en la Figura 4.3 del objeto *lemon*

5 Resultados

Una vez explicado el desarrollo llevado a cabo en la Sección 4, en este apartado vamos a comentar y mostrar los resultados obtenidos de forma visual, a modo de clarificar todas las salidas del dataset.

El dataset [1] forma parte del proyecto global y se ha subido a una carpeta en *Google Drive*[2] llamada How2Grasp, y la ordenación de los datos se puede ver a modo de ejemplo en la Figura 5.1. Hay una carpeta para cada objeto en la que se encuentra una subcarpeta para cada agarre, donde está la información de interés como son los contactos de los cinco dedos individualmente y la información del objeto y de la mano en un .txt.

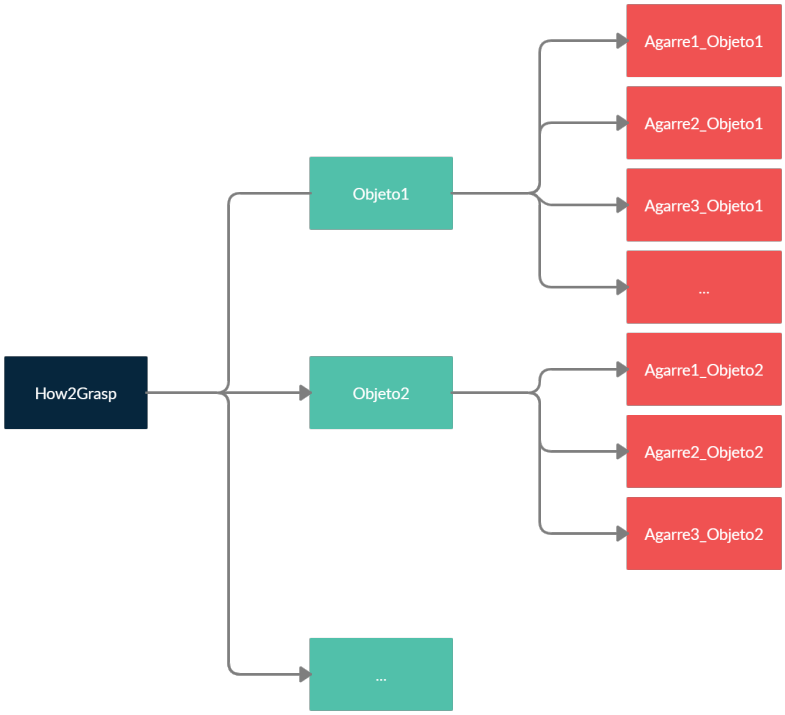


Figura 5.1: Diagrama ejemplo de carpetas del dataset *How2Grasp*.

En cuanto a los resultados obtenidos del script de Python en el que se obtienen los mapas de calor, se pueden guardar de tres formas distintas:

- En escala de grises (negro-blanco, como los hdr originales).

[1] https://github.com/3dperceptionlab/unrealhandgrasp/tree/dataset
[2] `https://drive.google.com/drive/folders/1iKhwNTr2xR4nGhLV--l7AFDqwdmVMlZe`

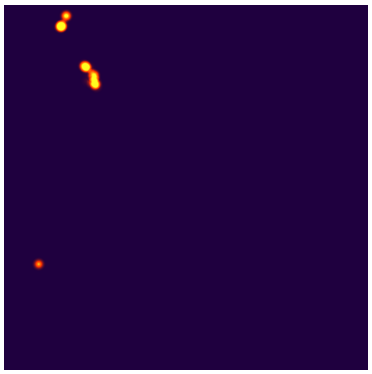
- Por colores, diferenciando cada uno de los dedos con un color.
- Como mapa de calor "sintético".

A su vez, los mapas de calor se pueden obtener tanto de los dedos individualmente como agregados, es decir, un mapa de calor único en el que se juntan los cinco dedos. Por lo que en cada uno de los tres modos para guardarlos se puede almacenar hasta un total de seis archivos, cinco texturas por dedo y otra del agregado. Aunque de UE4 se adquieren imágenes .hdr, y la salida del script las transforma a .png para una mayor compatibilidad con visores de imágenes comunes, incluso para poder mostrarse en documentos comunes creados mediante *Office Word* o con *latex*.

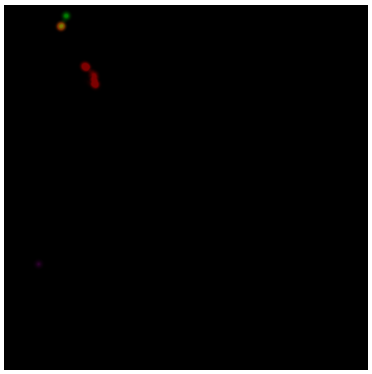
A continuación se van a mostrar ejemplos de los resultados obtenidos visualmente con diferentes objetos como son una lata de atún en la Figura 5.2, y una fresa en la Figura 5.3. Se pueden ver las texturas de los objetos, el mapa de calor agregado y el mapa de calor diferenciando con un dedo cada color. Además se añaden las texturas con los contactos añadidos desde diferentes vistas, tanto las originales como los mapas de calor. Examinando el agarre de la lata de atún encontramos que se ha cogido desde ambas caras con tres dedos, al igual que en la fresa, la cual se agarra con dos en la parte superior y uno en la inferior.

Una de las dificultades encontradas ha sido con objetos como la jarra que se puede observar en la Figura 5.4. Al tener superficies con espacios a la hora de agarrarla nos podemos encontrar con el impedimento de que los dedos colisionan con el objeto antes de llegar a pasar por el mango.

(a) Textura lata de atún

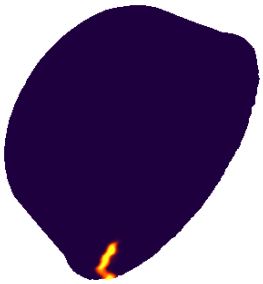


(b) Mapa de calor agregado

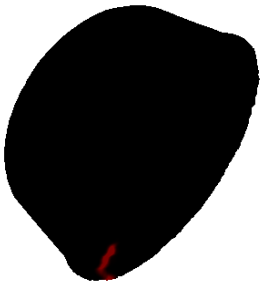


(c) Mapa de calor diferenciando dedos por colores

(d) Textura original vista 1

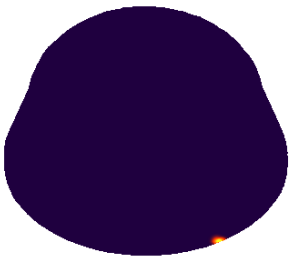


(e) Mapa de calor vista 1

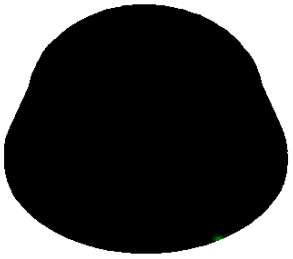


(f) Mapa de contactos por dedo 1

(g) Textura original vista 2



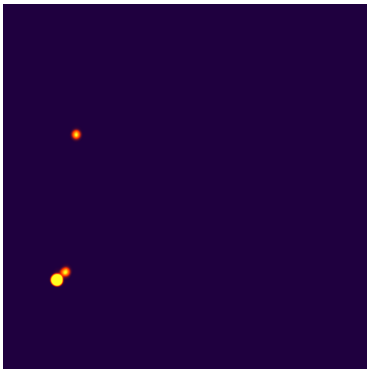
(h) Mapa de calor vista 2



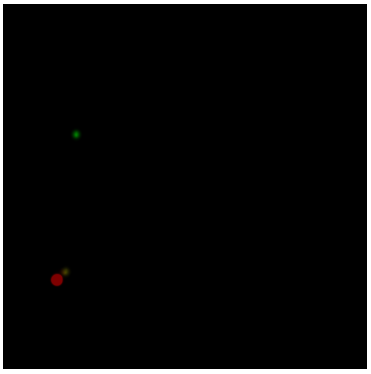
(i) Mapa de contactos por dedo 2

Figura 5.2: Lata atún

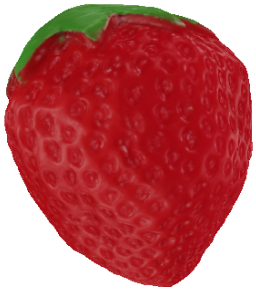
(a) Textura fresa



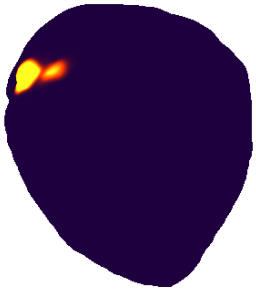
(b) Mapa de calor agregado



(c) Mapa de calor diferenciando dedos por colores



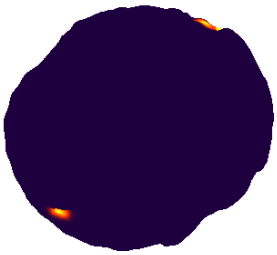
(d) Textura original vista 1



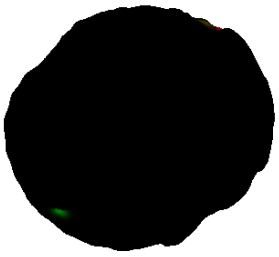
(e) Mapa de calor vista 1

(f) Mapa de contactos por dedo 1

(g) Textura original vista 2



(h) Mapa de calor vista 2



(i) Mapa de contactos por dedo 2

Figura 5.3: Fresa

(a) Puntos de contacto jarra

(b) Textura original jarra

Figura 5.4: Jarra

6 Conclusiones

Con el avance de la IA y la necesidad de grandes volúmenes de datos etiquetados para entrenar algoritmos de aprendizaje automático y profundo, hay campos en los que puede resultar difícil e incluso tedioso obtener un dataset con información relevante, es por esto que surge la necesidad de datos sintéticos, es decir, datos cuya información es obtenida a través de simulación del mundo real. Tras esto, con el objetivo de crear un dataset de agarres surge la duda de qué forma será la más cómoda para el usuario y obtenga un mejor resultado. Aquí es donde entran en juego los datos sintéticos y nace este proyecto, en nuestro caso han sido producidos a través de un entorno creado en UE4 y preparado especialmente para que la obtención y etiquetado de datos sea una tarea lo más sencilla y automatizada posible.

Puesto que el objetivo era crear un dataset de agarres, se necesitaban modelos de objetos 3D los cuales cargar en UE4 y así agarrar mediante las gafas de realidad virtual y el handtracking integrado, comentado en la Sección 3.3.1.1. Para esto se hacen uso de los modelos del dataset YCB.

Además, se han obtenido los contactos de los dedos individualmente, algo esencial para que el dataset tuviera toda la información de interés. Una vez obtenidos los puntos de contacto de cada uno de los dedos se ha llevado a cabo un script que es capaz de guardar el mapa de calor unificando y diferenciándolos.

Por tanto, podemos confirmar que se han llevado a cabo los objetivos principales del proyecto exitosamente al conformar un dataset sintético de agarres humanos en un motor gráfico como es UE4 haciendo uso de los objetos de YCB, además de ser capaces de diferenciar cada dedo individualmente, tarea importante marcada como objetivo esencial.

Todo lo que se ha realizado ha sido sustentado por una serie de proyectos previos mediante los cuales hemos sido introducidos en el mundo de motores gráficos, la realidad virtual y creación de datasets, con el añadido de programación en Python para llevar a cabo el mapa de calor.

Cabe destacar la utilidad de este dataset sintético que servirá para entrenar modelos de *Deep Learning*, con la esperanza de obtener buenos resultados en el mundo real haciendo uso de un mundo simulado.

6.1 Posibles mejoras y trabajo futuro

Tras haber explicado la totalidad del proyecto a lo largo de las secciones previas, se va a dedicar esta subsección a comentar las posibles mejoras de cara al futuro. Se van a tener en cuenta todo tipo de mejoras pensando no solo en mejorar el actual generador y dataset, sino en su uso como base para crear nuevos datasets de agarres.

- La cantidad de agarres por objeto es algo importante, es por esto que se han realizado un mínimo de 4 agarres por objeto utilizando varias manos. Pero otro de los parámetros importantes es la cantidad de personas que realizan los agarres que conforman el dataset.

Se propone por tanto la posibilidad de realizar agarres con diferentes personas a modo de obtener datos lo más genéricos posibles.

- Pensando en el campo de la robótica, sería interesante implementar algún tipo de extremo robótico y controlarlo bien sea mediante los mandos de las gafas de realidad virtual o mediante nuestras propias manos y realizar un dataset específico para esa herramienta.
- Este dataset está pensado para el agarre de objetos con una única mano, es decir, el tamaño de los objetos tiene cierta limitación. Surge así una posible mejora basada en la implementación de una lógica que permita el agarre de objetos con ambas manos y guarde información de todos los dedos diferenciando la mano a la que pertenecen.
- Comentado el tamaño en el punto previo, en relación a cualidades de los objetos, se propone una mejora en la que se tengan en cuenta propiedades físicas en el entorno de simulación para analizar más a fondo el agarre del objeto, a modo de añadir realismo.

Bibliografía

Akizuki, S., y Aoki, Y. (2019). Tactile logging for understanding plausible tool use based on human demonstration.. (Funding Information: This work was partially supported by JSPS KAKENHI Grant Number 17K12761. Publisher Copyright: © 2018. The copyright of this document resides with its authors. It may be distributed unchanged freely in print or electronic forms.; 29th British Machine Vision Conference, BMVC 2018 ; Conference date: 03-09-2018 Through 06-09-2018)

Brahmbhatt, S., Ham, C., Kemp, C. C., y Hays, J. (2019, 6). ContactDB: Analyzing and predicting grasp contact via thermal imaging. En *The ieee conference on computer vision and pattern recognition (cvpr).*

Brahmbhatt, S., Handa, A., Hays, J., y Fox, D. (2019). *Contactgrasp: Functional multi-finger grasp synthesis from contact.*

Calli, B., Walsman, A., Singh, A., Srinivasa, S., Abbeel, P., y Dollar, A. M. (2015). Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set. *IEEE Robotics Automation Magazine*, *22*(3), 36-52. doi: 10.1109/MRA.2015.2448951

Chao, Y.-W., Yang, W., Xiang, Y., Molchanov, P., Handa, A., Tremblay, J., … Fox, D. (2021). DexYCB: A benchmark for capturing hand grasping of objects. En *Ieee/cvf conference on computer vision and pattern recognition (cvpr).*

Chaurasia, G., Nieuwoudt, A., Ichim, A.-E., Szeliski, R., y Sorkine-Hornung, A. (2020, abril). Passthrough+: Real-time stereoscopic view synthesis for mobile mixed reality. *Proc. ACM Comput. Graph. Interact. Tech.*, *3*(1). Descargado de `https://doi.org/10.1145/3384540` doi: 10.1145/3384540

Depierre, A., Dellandréa, E., y Chen, L. (2018). *Jacquard: A large scale dataset for robotic grasp detection.*

Eppner, C., Mousavian, A., y Fox, D. (2020). *Acronym: A large-scale grasp dataset based on simulation.*

FERNÁNDEZ, Y. (s.f.). *El primer simulador vr de la historia.* Descargado 22/03/2018, de `https://www.xataka.com/historia-tecnologica/el-primer-simulador-vr-de-la-historia-tenia-forma-de-recreativa-y-se-invento-a-finales-de-los-50`

Garcia-Garcia, A., Martinez-Gonzalez, P., Oprea, S., Castro-Vargas, J. A., Orts-Escolano, S., Garcia-Rodriguez, J., y Jover-Alvarez, A. (2018). The robotrix: An extremely photorealistic and very-large-scale indoor dataset of sequences with robot trajectories and interactions. En *2018 ieee/rsj international conference on intelligent robots and systems (iros)* (p. 6790-6797). doi: 10.1109/IROS.2018.8594495

Han, S., Liu, B., Cabezas, R., Twigg, C. D., Zhang, P., Petkau, J., … Wang, R. (2020, julio). Megatrack: Monochrome egocentric articulated hand-tracking for virtual reality. *ACM Trans. Graph.*, *39*(4). Descargado de https://doi.org/10.1145/3386569.3392452 doi: 10.1145/3386569.3392452

Hodaň, T., Michel, F., Brachmann, E., Kehl, W., Glent Buch, A., Kraft, D., … Rother, C. (2018). BOP: Benchmark for 6D object pose estimation. *European Conference on Computer Vision (ECCV)*.

Liang, H., Ma, X., Li, S., Görner, M., Tang, S., Fang, B., … Zhang, J. (2018). Pointnetgpd: Detecting grasp configurations from point sets. *CoRR*, *abs/1809.06267*. Descargado de http://arxiv.org/abs/1809.06267

Mahler, J., Matl, M., Satish, V., Danielczuk, M., DeRose, B., McKinley, S., y Goldberg, K. (2019). Learning ambidextrous robot grasping policies. *Science Robotics*, *4*(26), eaau4984.

Martinez-Gonzalez, P., Oprea, S., Castro-Vargas, J. A., Garcia-Garcia, A., Orts-Escolano, S., Garcia-Rodriguez, J., y Vincze, M. (2021). Unrealrox+: An improved tool for acquiring synthetic data from virtual 3d environments. *CoRR*, *abs/2104.11776*.

Martinez-Gonzalez, P., Oprea, S., Garcia-Garcia, A., Jover-Alvarez, A., Orts-Escolano, S., y Garcia-Rodriguez, J. (2019). UnrealROX: An extremely photorealistic virtual reality environment for robotics simulations and synthetic data generation. *Virtual Reality*. Descargado de https://doi.org/10.1007/s10055-019-00399-5 doi: 10.1007/s10055-019-00399-5

Mulero Pérez, D. (2021). *Interacción con objetos en realidad virtual utilizando tracking de manos.* Repositorio Institucional de la Universidad de Alicante (RUA). Descargado de http://hdl.handle.net/10045/115937

Oprea, S., Martinez-Gonzalez, P., Garcia-Garcia, A., Castro-Vargas, J. A., Orts-Escolano, S., y Garcia-Rodriguez, J. (2019). A visually realistic grasping system for object manipulation and interaction in virtual reality environments. *Computers & Graphics*, *83*, 77 - 86. Descargado de http://www.sciencedirect.com/science/article/pii/S0097849319301098 doi: https://doi.org/10.1016/j.cag.2019.07.003

Upton, M. (s.f.). *Toyota human support robot: What is it and how can it be used?* Descargado 30/03/2021, de https://mag.toyota.co.uk/toyota-human-support-robot/

Zapatero, N. (s.f.). *Diferencias entre la realidad virtual y la realidad aumentada.* Descargado 03/09/2017, de https://www.turiskopio.com/5-diferencias-entre-la-realidad-virtual-y-la-realidad-aumentada

¿qué son los datasets y dónde conseguirlos? (s.f.). Descargado de https://keepcoding.io/blog/que-son-datasets/